Fermilab/BD/TEV
Beams-doc-1276
February 4, 2005
Version 03

Tevatron Beam Position Monitor Front End Software User's Guide

DRAFT DRAFT DRAFT

Margaret Votava, Luciano Piccoli, Dehong Zhang, Kurt Biery
Fermilab, Computing Division, CEPA

Abstract

This document is geared to help past developers remember what they did, future developers to know how to build and distribute the software, commissioners to understand how to configure a system, and maintainers to help diagnose software problems. Contact tev-bpm-daqsw@fnal.gov for comments/updates.

1 Overview

This document is intended to be a guide for the front end users of the Tevatron BPM system. It is meant to help

- developers build system and verify the builds
- commissioners now how install and commission boards and crates
- maintainers diagnose problems

2 Commissioning

2.1 Crate Controller Configuration

The crate controller in a TBPM house is a Motorola 2400 running vxWorks. Node names are selected with the convention of tbpm<id> where <id> is a two character identifier of location in the ring, e.g., tbpma3.

2.1.1 Registration

Generic instructions can be found at:

http://www-bd.fnal.gov/controls/micro_p/rom_install.html.

IP addresses must be obtained through the Acceleration Division networking group. Before requesting an IP address, you must know the Fermilab ID and Ethernet MAC address of the board. Once that has been identified, register the board at:

http://www-bdnew.fnal.gov/Netwebrequests/net-connection.asp

Once the IP address is obtained, each host will also need an ACNET address, which consists of the above node name plus an assigned trunk and node id. Contact Brian Hendricks to procure a trunk and node id.

2.1.2 Boot Parameters

Boards are currently running VxWorks v5.5. Each crate controller should have the following boot parameters:

```
boot device          : dc0
processor number     : 0
host name            : fecode-bd
file name            : vxworks_boot/kernel/mv2400-512MB/vxWorks-a32-512
inet on ethernet (e) : 131.225.<xxx>.<yyy>:ffffff00
inet on backplane (b):
host inet (h)        : 131.225.121.145
gateway inet (g)     : 131.225.<xxx>.200
user (u)             : vxworks_boot
ftp password (pw) (blank = use rsh):
flags (f)            : 0x0
target name (tn)     : <nodename>_0x<trunkid><nodeid>
startup script (s)   : vxworks_boot/fe/tpm/tpmstartup
other (o)            :
```

Where <xxx>, <yyy>, <trunkid>, and <nodeid> are obtained in the registration step above .

Note that this is a custom kernel – it specifically adds the gateway node to the route table with the following command:

```
routeAdd "0.0.0.0", "131.225.<xxx>.200"
```

This is done so that all houses can use the same startup script.

2.1.3 Startup Script

The Tevatron BPM software is designed to autodetect hardware and addresses at run time, so all BPMs can share the same startup script. The startup script pointed to in the boot parameters currently contains:

```
#-----------------------------------------------------------------------
#--- General initialization
#-----------------------------------------------------------------------
shellPromptSet( "Booting incomplete->" )
taskPrioritySet( taskIdFigure( "tExcTask" ), 1 )
taskPrioritySet( taskIdFigure( "tLogTask" ), 250 )
# get target name in 6 characters
targetName=malloc(20)
gethostname(targetName,20)
targetName[6]=0
#routeAdd ("239.128.1.1","131.225.126.200")
#-----------------------------------------------------------------------
# mount NFS file systems - 1217, 5143 is user vxworks_boot group bdmicrop
#-----------------------------------------------------------------------
topLevel="vxworks_write/fe/tbpm"
outputDir=malloc (strlen(topLevel) + strlen(targetName))
strcpy(outputDir,topLevel)
strcat(outputDir,targetName)
nfsMount( "fecode-bd", "vxworks_boot/module/PPC604", "/controls"  )
nfsMount( "fecode-bd", "vxworks_boot/fe", "/fe"  )
nfsMount( "fecode-bd", outputDir, "/write" )
nfsAuthUnixSet( "fecode-bd", 1217, 5143, 0, 0 )

# TEMPORARY
nfsMount( "fecode-bd", "vxworks_write/fe/fccts1", "/write1")

#-----------------------------------------------------------------------
#--- Load MOOC and ACNET services
# THESE LOCATIONS NEEED TO CHANGE!!!!!
#----------------------------------------------------------------------
# class library to initialize the techno box
ld (1, 1, "vxworks_boot/fe/rbpm/controls/pmcclassLib_mv2400-lastest.o");
# mooc

# 5.4 libraries
#ld < /controls/libpmctrig-latest.o
#ld < /controls/libssm-2.3.o
#ld < /controls/acnet-1.021.o
#ld < /controls/libmooc-3.6.o

# 5.5 libraries
ld < /controls/libpmctrig-2.13.o
ld < /controls/libssm-2.4.o
ld < /controls/acnet-1.026.o
ld < /controls/libmooc-3.7.o
#ld < /controls/libmooc-3.6.o
#ld < /controls/libmooc-latest.o

ACNET_SSM_avail=0

# what is this?
ld < /fe/rfiinst/devlib/VW_54/MVME2434/libmiscutil.out
# bpm stuff
ld < /fe/echotek_tbpm/libechotek_tbpm.out
# Warren's ECSG-1R3ADC-PMC I/O drivers
ld < /fe/rfiinst/lib/VW_54/MVME2434/libecsg1r3adcfermi.out
ld < /fe/tbpm/libculite.out
ld < /fe/gbpm/libgbpm.out
ld < /fe/tbpm/libtbpm.out


#-----------------------------------------------------------------------
#--- Start things
# THESE LOCATIONS NEEED TO CHANGE!!!!!
#-----------------------------------------------------------------------
# set the downloadable file
set_rbf_file ("vxworks_boot/fe/rbpm/controls/pmcucd.rbf");
# download the techno box; 0 is the instance or slot
pmcucd_start(0)
```

```
# set the addresses for ucd
pmcucd_set_adr (get_alteramem(0))
pmcucd_set_madr (get_alteramem(0)+0x1000)

initssc (0,calloc(0x10000,1))
MoocNew()

# void ecsg( int fref, int n, int p, int r, int b, int a, int c, int d );
# request 73.5 MHz and get 73.444444 MHz
#ecsg( 53104696, 504, 8, 40, 63, 0, 3, 3)

cd( "/fe/tbpm/ini" )
ecdr814gcReadSetup("53MHzNarrowBand.ini",    0);
ecdr814gcReadSetup("53MHzEnsemble.ini",      1);
ecdr814gcReadSetup("53MHzEnsemble-debug.ini",2);
#ecdr814gcReadSetup("rawDmaD64.ini",          3);
#ecdr814gcReadSetup("rawDmaD64.ini",          4);
#ecdr814gcReadSetup("rawDmaD64.ini",          5);
#ecdr814gcReadSetup("rawDmaD64.ini",          6);
ecdr814gcReadSetup("rawDmaD64.ini",          7);
ecdr814gcReadSetup("rawDmaD64Long.ini",      8);
ecdr814gcReadSetup("count.ini",             19);
ecdr814gcShowSetupAll();

# FilterSetIndex( 0, 23 )
# FilterSetIndex( 2, 23 )

#$$$MoocAlarmScan( 1 )
MoocDeviceDownLoad( 0 )

#------------------------------------------------------------------------
#--- Start trace buffer
#------------------------------------------------------------------------
# Clock prescale value
*0xfef80020=0x63ce0001
traceInit 100000,10,0xfef80100
traceMode 1
traceGlobalOn 0,25,0
traceOff 1,0,31,0

#traceMode 2
#traceGlobalOn 0,31,2
#traceOff 1,0,31,2
traceGlobalOff 17,17,0
traceGlobalOff 19,19,0
traceGlobalOff 20,20,0

#------------------------------------------------------------------------
#--- Finish up
#------------------------------------------------------------------------
prompt = malloc(30)
strcpy (prompt,targetName)
strcat (prompt,":tbpm> ")
shellPromptSet ( prompt )
free(prompt)
free(targetName)
free(outputDir)

traceOthersOff

#ecdr814gcInstall("ECDR-GC814-
FV2",0x9000,0x19000000,0,"/config/rcvr_stratix_dual_14b.rbf",4)
#ecdr814gcDemo_count( "/write/Count_Test_0x9000", 1, 0, 0, 0, 100 )

TBPMSetPriority ("ucd60hz", 40);
TBPMSetPriority ("tExcTask", 40);
TBPMSetPriority ("tWdbTask", 40);

# Change the priority of the tShell task
TBPMChangeShellPriority (50);

#traceGlobalOff 12,12,0
#traceGlobalOff 16,16,0
traceGlobalOff 15,16,0

bpmStart 500

a_alarms_enable ()
```

```
a_reload_data ()
```

2.1.4 Changing the vxWorks startup script for all houses

The tevatron BPM houses share a common startup script that resides in nova.fnal.gov:/fecodde-bd/vxworks_boot/fe/tbpm; however, it is maintained in the tbpm CVS module. If the startup script for all houses need to change, please follow the build instructions in subsequent section.

Note that we are loading fixed versions of the ACNET/MOOC libraries. These versions need to match the versions of ACNET/MOOC header files that are compiled into both gbpm and tbpm. When reloading a different version of mooc, you will also need to edit the _acnetheaders.h file in both packages and recompile (see build instructions below).

2.1.5 Changing the vxWorks startup script for a particular house

While debugging problems at a specific house, there will sometimes be a need to have a startup script that is specific to that particular house. Since this is only a temporary state, there is no need to track the changes. Simply:

```
nova> cd /fecode-bd/vxworks_boot/fe/tbpm
nova> cp tbpmstartup temp_startup
```

Edit the temp_startup script as necessary, being sure to change the vxworks boot parameters to use this script. Please remember to reset the boot parameters to use the canonical script when done.

2.2 Echotek Board Configuration

The current echotek driver supports up to 20 different setup files that are specified in the startup script. Different configurations are used based on the mode of operation of the software. See document #860 for a description of the operational modes. In a particular mode, all echotech modules in a house will use the same setup file, ie there is no provision for allowing the boards to be configured differently. By convention, these setup files have an extension of .ini.

Additionally, each board setup file contains another pointer to a file that contains information regarding a channel setup. Unless you really know what you're doing, all channels should be configured with the same file. By convention, the channel setup files have an extension of .ch. By convention, the <abc>.ini file will read in a channel file named <abc>.ch.
The driver reads the associated files from the current working directory, so it's important to cd to this directory in the vxWorks startup script.

Operational Mode	**Setup file index**
Closed orbit	0
Turn X Turn	1
First injection	1

Raw ADC	7
Closed orbit debug	2

2.3 Timing Module Configuration

2.3.1 Recycler Timing Board

Additional software that needs to be added to the startup script to run the recycler timing board are:

```
# class library to initialize the techno box
# only needed for
ld (1, 1, "vxworks_boot/fe/rbpm/controls/pmcclassLib_mv2400-lastest.o");
# void ecsg( int fref, int n, int p, int r, int b, int a, int c, int d );
# request 73.5 MHz and get 73.444444 MHz
ecsg( 53104696, 504, 8, 40, 63, 0, 3, 3)
```

3 Developing

There are 3 packages that constitute the tbpm software, they are:

- gbpm – generic bpm classes
- tbpm – bpm implementation specific to the tevatron
- echotek_tbpm – underlying echotek driver

3.1 Build Environment

The Tevatron BPM software uses the RFI build methodology. Please refer to Beams Document #1271 for a description of the tools and, for first time users, how to configure your account on nova.fnal.gov. Following these instructions, please set your work group to rfiinst. All code is compiled on nova.fnal.gov.

3.2 Build Instructions

All packages are built in a similar fashion

```
      nova-> cd ~/esd/src             # only do once; create if needed
      nova-> cvs checkout gbpm        # only do this once
      nova-> setup gbpm
      nova-> cvs update -d            # cvs checkout gbpm if first time
      nova-> make clean
      nova-> make
      nova-> make development         # put in development location
      nova-> make test                # put in test location
      nova-> make production          # put in production location (will
                                      # be installed at next reboot)
Library locations are:

nova.fnal.gov:/fecode-bd/vxworks_boot/fe/gpm/devgpm.out  # development
nova.fnal.gov:/fecode-bd/vxworks_boot/fe/gpm/testgpm.out # test
nova.fnal.gov:/fecode-bd/vxworks_boot/fe/gpm/libgpm.out  # production
```

There is a similar set of commands to build tbpm and echotek_tbpm.

3.3 Unit Tests

The three tbpm packages have unit or module tests associated with them. The unit tests are designed to verify the core functionality of the module and are designed to be as standalone as possible. The gbpm and tbpm packages use an underlying software package called culite, which is a Computing Division maintained product that is built remotely and then distributed to nova. It is installed in

nova.fnal.gov:/fecode-bd/vxworks_boot/fe/culite

Because the unit tests in and of themselves are large, they should not be loaded when running in a production mode and are meant primarily for release verification. Running the unit tests for gbpm and tbpm involves the following steps:

1. Build the package with unit tests turned on. Put the result in the "test" library:
   a. edit the project Makefile and add '-DUNIT_TEST' to the assignment of the DEFINES variable
   b. 'make test'

2. Load the unit test software:
   a. Create a startup script for unit testing, which includes loading unit test helper library along with the test libraries just built

```
ld < /fe/tbpm/libculite.out
```

3. Run the Unit tests. Each package has a self-contained unit test which calls all individual module tests. To run all the unit tests, from the vxworks shell:

```
vxworks-> GBPMUnitTest
vxworks-> TBPMUnitTest
```

4. For reference, the definitions of the CULITE_UNIT_TEST, CHECK, and other preprocessor macros can be found in the culite Test.h header file.

It is currently not possible to build the echotek_tbpm library without the unit tests. To run these unit tests, from the vxworks shell:

```
vxworks-> ecdr814gcUnitTestUnitTest(a16a, setupIndex)
```

where a16a is the address of the card to test (one of 0x8000, 0x9000, …, 0xf000, or 0 to test all cards) and setupIndex is the setup file index described in section 2.2.

3.3.1 Echotek Driver Unit Tests

All of the testing is contained in the ecdr814gcUnitTestUnitTest function. It requires Echotek cards to be present in the crate at the requested address(es).

3.3.2 GBPM Unit Tests

No test requires any more hardware than the crate controller except where otherwise noted.

```
  void DataEntryUnitTest (int initTrace);
  void TestDataEntryUnitTest ();
  void PackStrategyUnitTest ();
  void TestPackerUnitTest ();
  void EventUnitTest ();
  void EventListenerUnitTest ();
  void EventGeneratorUnitTest ();
  void InterruptEventGeneratorUnitTest (int initTrace);
  void HardwareInterruptEventGeneratorUnitTest (int initTrace);
  void TimeEventGeneratorUnitTest (int initTrace);
  void AuxTimeEventGeneratorUnitTest (int initTrace);
  void TCLKEventGeneratorUnitTest ();     // requires timing board
  void AlarmGeneratorUnitTest ();
  void DataAcquisitionTaskUnitTest (int initTrace);
  void DataBufferUnitTest (int initTrace);
  void CircularDataBufferUnitTest (int initTrace = 1);
  void TestControlUnitTest (int initTrace);
  void TestControlTaskUnitTest (int initTrace);
  void AlarmTaskUnitTest (int initTrace);
  void ConfigManagerUnitTest (int initTrace);
  void PackerUnitTest ();
  void ConfigStateChangeListenerUnitTest (int initTrace);
  void StateChangeEventGeneratorUnitTest (int initTrace);
  void ConfigStateChangeTaskUnitTest (int initTrace);
```

3.3.3 TBPM Unit Tests

These tests are not currently fully functional. The following tests are available, but only the first four are called from TBPMUnitTest. And some of the tests may not run if there is no other hardware in the crate other than the crate controller.

```
  void TBPMClosedOrbitPackerUnitTest (int initTrace);
  void TBPMTurnByTurnPackerUnitTest (int initTrace);
  void TBPMDiagnosticSystemUnitTest (int initTrace);
  void EchoTekPoolUnitTest (int initTrace);
  void TBPMControlUnitTest (int initTrace);
  void TBPMRawClosedOrbitPackerUnitTest (int initTrace);
  void TBPMBufferReadoutUnitTest (int initTrace);
```

4 Maintaining

How/when to change .ini and .ch files. Tune timing delays.
Not written yet.

5 Diagnostics

Once the system is up and running, how do you start analyzing problems? There are many diagnostic tools available. Choosing the right one to most expediently analyze the problem is a combination of experience and instinct. The following sections describe various diagnostic information that is available in the tbpm software.

5.1 Software Statistics

The bpm software internally keeps structures that one can peek at through the vxworks shell. The bpmHelp command will list the most current set of commands that are available:

```
fccts1:tbpm-dev> bpmHelp
---------------------------------------------------------------
bpmStart (rate, source, useTCLK, bsyncBase)
Description: Starts up the TBPM system
Parameters: rate - clock generator rate (in Hz)
            source - 0 for EchoTek boards
                     1 for TestEchoTek(don't use EchoTek board)
            useTCLK - 0 for using TCLK decoder
                      1 for ignoring TCLKs (don't use decoder)
            bsyncBase - base address for the bsync decoder

bpmStartTest (rate)
Description: Starts the the TBPM system using no hardware
Parameters: rate - fast abort frequency (in Hz)

bpmPeek (bpm, buffer)
Description: Shows the latest entry of a selected buffer
Parameters: bpm - 0 through 11
            buffer - fast abort = 11
                   - slow abort = 12
                   - profile    = 13
                   - display    = 14
                   - user TCLK  = 15
                   - tbt        = 16
                   - inj. tbt   = 17
                   - inj. c.o.  = 18
                   - diagnostic = 19
                   - closed orb.= 20

bpmPeekPos (bpm, buffer, position, format)
Description: Same as bpmPeek, element position can be specified
Parameters: position - element index within the buffer
            format - 1: for proton full information
            format - 2: for pbar full information

bpmShow ()
Description: Display general system information

bpmShowTasks ()
Description: Display status of the data acquisition tasks

bpmShowGenerators ()
Description: Display status of the event generators

bpmShowQueues ()
Description: Display status of event queues

bpmShowBuffers ()
Description: Display current status/info of system buffers

bpmShowConfig ()
Description: Display detailed system configuration

bpmShowModes ()
Description: Show the last N modes of operation of the system

bpmShowTCLK ()
Description: Display the TCLK history

bpmShowAlarms (enabled)
Description: Show current alarms in the system
Parameters: enabled - 0 for enabled alarms
                    - 1 for all alarms
```

```
bpmClearBuffers
Description: Clear all buffers in the system

bpmPeekDiag ()
Description: Display current contents of the diagnostic buffer

bpmDumpBuffer (bpm, buffer)
Description: Display all contents of the buffer for a given bpm
             The position/intensities and I/Q pairs are shown
Parameters: bpm - 0 through 11
            buffer - buffer index (see bpmPeek help)


---------------- Development/Test functions -------------------
bpmMode (mode, turns)
Description: Change system the mode of operation
Parameters: New mode of operation
            No parameters or mode == 0 for list of modes.
            turns - number of turns in turn-by-turn mode
bpmTCLK (tclk)
Description: Generate a TCLK event that is caught by the system
Parameters: tclk - 0x47: ABORT
                   0x4D: Tevatron reset - proton injection
                   0x71: Prepare for beam
                   0x75: BPM profile
                   0x77: Arm turn-by-turn measurement
                   0x78: BPM display
                   0x100: User TCLK generation

bpmTBT (only in test mode)
Description: Generate a turn-by-turn measurement. If in
             injection mode take an injection turn-by-turn
             measurement

bpmTBTEvent (tclk)
Description: Change the TCLK event that arms a turn-by-turn
             measurement.
Parameters: tclk - TCLK that will arm the turn-by-turn

bpmChangeConfig (index, value)
Description: Assign a new value to the integer variable at the
             given index.
Parameters: index - entry number in the config manager to be changed
            value - value to be assigned to the entry


---------------------------------------------------------------
```

5.2 Trace Diagnostics

The Tevatron BPM software packages are using a trace facility to monitor software behavior. This document is not meant to be a tutorial for that facility - Refer to the output of the traceHelp command (sample below) for an overview of the trace commands that are available.

```
fccts1:tbpm-dev> traceHelp
Trace Facility Help

traceMode(mode)          Sets the target(s) of the trace messages.
                         'mode' can be:
                             0 = tracing is disabled
                             1 = trace into circular buffer
                             2 = trace to VxWorks log facility
                             3 = trace to both

traceModeGet()           Returns the current trace mode.

traceInfo()              Shows information associated with the
                         trace facility. Each trace source is
                         also shown along with its level mask.

traceShow(opts, lines, skip, fd)
                         Dumps the contents of the trace buffer.
                         The arguments have the following effects:
                         'opts' is a bit mask:
                           bits 0,1:
                             0 = Relative timestamps are shown
                             1 = Absolute, but zeroed, timestamps
                             2 = Absolute timestamps
                           bit 2:
                             0 = do not show trace level
                             1 = show trace level
                         'lines' can be:
                             0 = Display all entries (default)
                                 Otherwise only display the first
                                 'lines' entries
                         'skip' can be:
                             0 = Don't skip entries
                                 Otherwise display every 'skip'
                                 entry
                         'fd' can be an open file descriptor. If
                         it is zero, then standard output is used.

traceReset()              Clears the circular trace buffer.

traceOn(id, lo, hi, f)       Turns on the specified levels of trace
                         source 'id'. This value can be found
                         using traceInfo(). 'lo' and 'hi' define
                         the range of levels to enable (0 - 31).
                         f is the "function" (currently -
                         0=circular, 1=logMsg.

traceOff(id, lo, hi, f)      Turns off the specified levels of trace
                         source 'id'. 'lo' and 'hi' define
                         the range of levels to disable (0 - 31).
                         f is same as above.

traceGlobalOn(lo, hi, f)     Like traceOn(), but it affects every
                         trace source.
                         f is same as above.

traceGlobalOff(lo, hi, f)    Like traceOff(), but it affects every
                         trace source.
                         f is same as above.

traceFreezeOn(lo, hi)     Enables a range of "freeze" levels. If
                         your code calls traceFreeze(lvl) and the
                         lvl'th freeze bit was set, the circular
                         trace queue is frozen (similar to calling
                         traceMode(0).) traceInfo() shows the
                         current freeze mask.

traceFreezeOff(lo, hi)    Disables a range of "freeze" levels.
```

As a quick overview, we typically use the tracing facility in a circular buffer mode where all traced processes write into that buffer. The software is exploiting many levels of trace that can be turned on/off to fine tune which information is stored in the buffer.

As an example, the following use of the traceShow command would display (on standard output) the most recent 100 entries in the trace buffer without skipping any entries, and each entry would be shown with an absolute timestamp and its trace level.

```
      traceShow 6, 100, 0, 0
```

5.3 VME Backplane Diagnostics

Bus analyzers are a powerful tool in understanding what is actually happening on the backplane. Using a VMETRO xdkjldsf, the following traces where obtained to show vme timing of typical transfers: xdkjldsf == VBT 320 or VBT321 ?

Missing figures

Figure x. Closed Orbit Measurement

Figure x. TurnByTurn Measurement

Figure x. First Turn Measurement

5.4 Looking at ACNET variables

From an ACNET console page, select the "ACL Edit/Run" option from the Pgm_Tools menu which is located near the top right corner of the page. In the Action section of the ACL Edit Window, type 'read <variableName>' and type <ctrl>-r to run the command. When the command is run, the ACNET window will split into two parts, the command source (which you typed) and the output. As an example, 'read V:PING' followed by <ctrl-r> should return a result like "V:PING = 5   ping" in the output pane.

Setting ACNET variables uses the same ACL Edit/Run application, but first you need to configure your ACNET console to allow variables to be set. This is done from the Utilities window. Click on the Setting option in the Utilities window and set the value to one hour. Back on the ACL Edit/Run page, you can now set variable values using commands of the form 'set <variableName>=<value>'. For example, to set the value of V:PING, you could use a command like 'set V:PING=5' (followed by <ctrl>-r).

6 Test Stand

6.1 Test Crate Modules

Needs completion

7 Appendix: Node name/ACNET addresses

node name	ip address	ACNET address	Location
Fccts0	131.225.126.243		FCC3
Fccts1	131.225.126.234	0x0bd4	FCC3
TBPMA3	131.225.127.233	0x0c2b	A3
TBPMB3	131.225.127.228		

Need completion

configuration files

acnet devices

labview board testing code